Author

Name: Satyam Maurya
Roll number: 23F1001514
Email: 23f1001514@ds.study.iitm.ac.in
I am a pre-final year B.Tech student at NIT Hamirpur and a BS student in Data Science and Applications at IIT Madras. I am passionate about leveraging my technical expertise to tackle real-world challenges and constantly strive to enhance my skills through hands-on projects.

Description

This project, Quiz Master, is designed as a web-based platform to manage quizzes efficiently. The system allows users to register, take quizzes, and track their scores, while admin can create and manage quizzes and users. The primary goal is to create an interactive and user-friendly quiz system with a structured backend

Technologies used

- **Python:** The core programming language used for implementing logic, handling data, and managing routes.
- **Flask (Web Framework):** A lightweight and flexible framework for building web applications.
- **Flask-SQLAlchemy (ORM):** Enables database management using Python objects instead of raw SQL queries, enhancing readability and security by reducing SQL injection risks.
- **Flask-Bcrypt (Password Hashing):** Ensures secure password storage by hashing passwords before saving them to the database.
- **Flask-WTF:** Facilitates form handling and validation within the application.
- **SQLite / MySQL:** Used as the database to efficiently store and manage quiz-related data.
- **Jinja2 (Templating Engine):** Enables dynamic content rendering within HTML templates.
- **HTML5:** Provides the structural foundation for web pages.
- **CSS & JavaScript:** Enhances the frontend by adding styling and interactivity.
- **Bootstrap 5 (CSS Framework):** Simplifies the styling process with pre-built classes, ensuring a consistent design and mobile responsiveness.
- **Git & GitHub:** Used for version control, collaboration, and project tracking.

Architecture

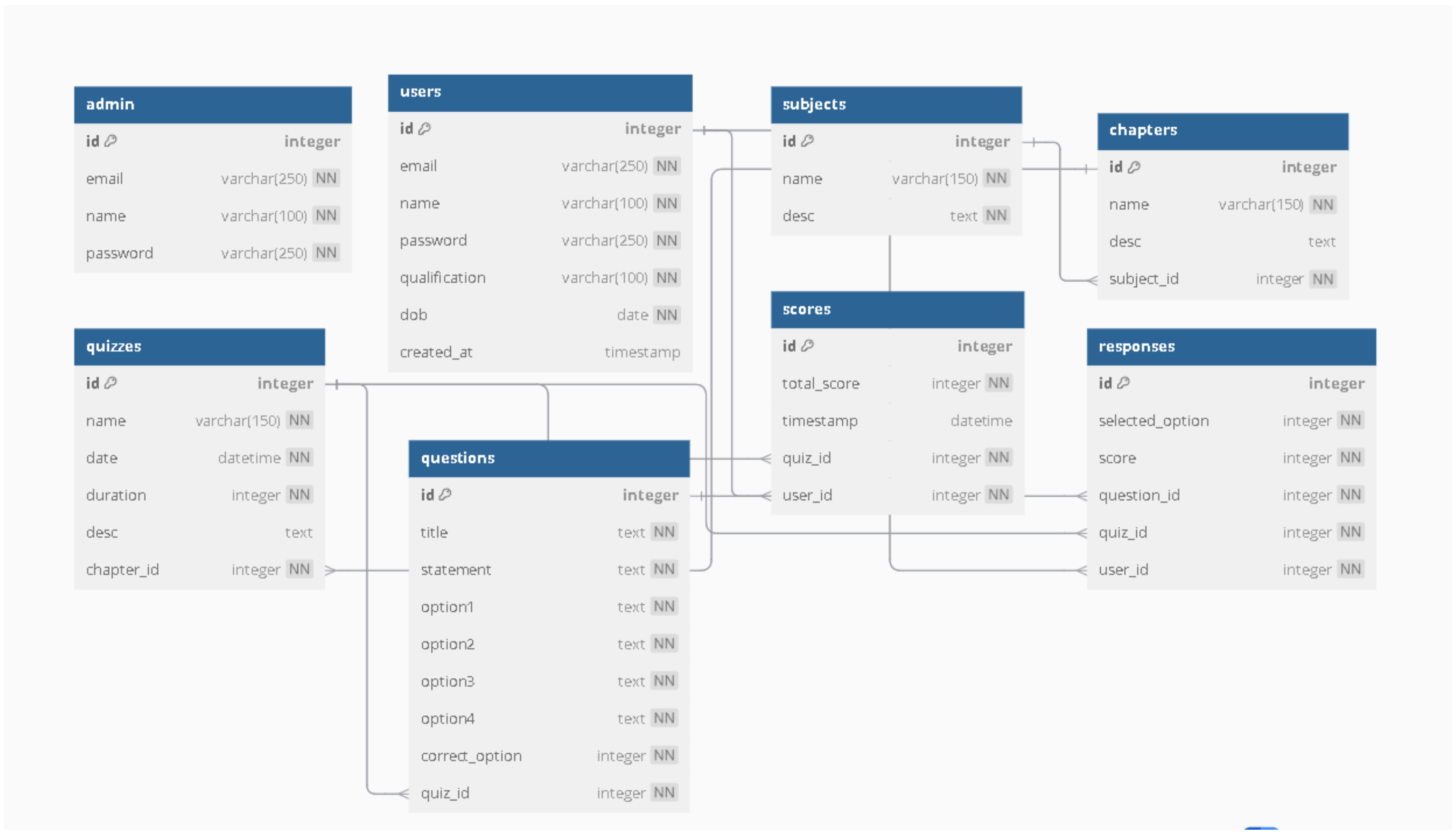
- **app.py:** The main entry point of the application, responsible for initializing the Flask app, configuring routes, and running the server.
- **Controllers (controllers/):** This directory contains the core application logic and handles API requests.
    - **routes.py:** Defines the API endpoints and maps URLs to appropriate functions.

    - **appFunctions.py:** Contains helper functions for quiz-related operations.
    - **forms.py:** Manages form validation and user input processing.
- **Models (models/):** This directory holds the database schema and ORM models.
    - **model.py:** Defines the database models using SQLAlchemy.
    - **__init__.py:** Initializes the database connection and configurations.
- **Instance (instance/):** Contains the SQLite database file (database.db), where all quiz-related data is stored.
- **Migrations (migrations/):** Facilitates database version control and schema migrations using Flask-Migrate.
- **Static Files (static/):** Stores frontend assets like stylesheets, JavaScript files, and images.
    - **script.js:** Manages client-side interactions and dynamic UI elements.
    - **styles.css:** Defines the layout and design of the web pages.
    - **images/:** Contains static images used in the application.
- **Templates (templates/):** Holds the HTML templates for rendering the web pages dynamically using Jinja2.
- **Virtual Environment (quizenv/):** Contains all the necessary dependencies and environment settings for running the Flask application.
- Configuration and Documentation Files:
    - **.gitignore:** Specifies which files should be ignored by version control (Git).
    - **README.md:** Provides a summary of the project, setup instructions, and usage details.
    - **requirements.txt:** Lists all dependencies required for the project.

Features Implemented

- **User Authentication:** Secure login and registration system using Flask-Bcrypt for password hashing.
- **Quiz Management:** Administrators can create, edit, and delete quizzes.
- **Question Handling:** The system supports multiple-choice and descriptive questions.
- **Real-time Scoring:** User responses are evaluated automatically, and scores are recorded instantly.
- **Performance Tracking:** Users can view their past quiz attempts and track their progress.
- **Admin Dashboard:** A structured interface for administrators to manage quizzes, users, and responses.
- **Bootstrap UI:** Enhances user experience by providing a responsive and visually appealing design.
- **Database Storage:** Uses SQLite for efficient and lightweight data management.
- **Role-based Access Control:** Differentiates between user and admin functionalities, restricting unauthorized access.

DB Schema Design



admin	
id	integer
email	varchar(250) NN
name	varchar(100) NN
password	varchar(250) NN

users	
id	integer
email	varchar(250) NN
name	varchar(100) NN
password	varchar(250) NN
qualification	varchar(100) NN
dob	date NN
created_at	timestamp

subjects	
id	integer
name	varchar(150) NN
desc	text NN

chapters	
id	integer
name	varchar(150) NN
desc	text
subject_id	integer NN

quizzes	
id	integer
name	varchar(150) NN
date	datetime NN
duration	integer NN
desc	text
chapter_id	integer NN

questions	
id	integer
title	text NN
statement	text NN
option1	text NN
option2	text NN
option3	text NN
option4	text NN
correct_option	integer NN
quiz_id	integer NN

scores	
id	integer
total_score	integer NN
timestamp	datetime
quiz_id	integer NN
user_id	integer NN

responses	
id	integer
selected_option	integer NN
score	integer NN
question_id	integer NN
quiz_id	integer NN
user_id	integer NN

Video

https://drive.google.com/file/d/1vI81Y5UsciGLPm0SB6VSmKdXbkaz8UsI/view?usp=sharing